

ClassConnect

Students: Obed Turnier, Mathew Lezcano, Matthew Ambroise

Instructor/Faculty: *Masoud Sajadi*, Florida International University

PROBLEM

Students in university courses often struggle to get timely help on assignments, concepts, and projects. Traditional communication methods such as email, group chats, or discussion boards are often disorganized, lack engagement, and do not prioritize the most important questions. This leads to delayed responses, duplicated questions, and reduced collaboration between students and instructors.

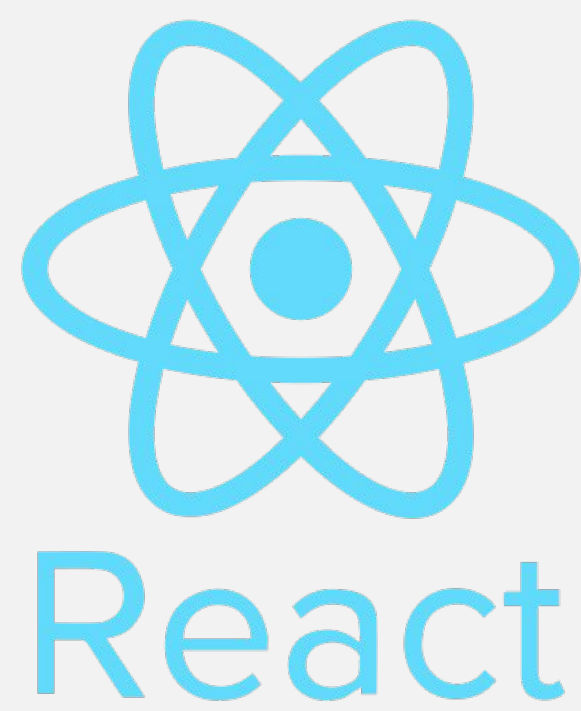
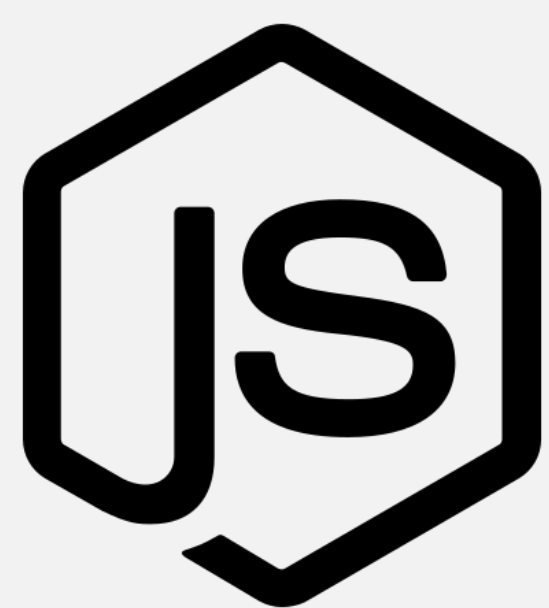
CURRENT SYSTEM

Existing systems such as Canvas discussion boards and group messaging platforms provide basic communication tools but lack real-time interaction, structured prioritization, and effective engagement mechanisms. Questions are not ranked by importance, responses are not centralized, and there is limited visibility into unresolved or critical issues. These limitations reduce efficiency and make it difficult for students and instructors to collaborate effectively.

REQUIREMENTS

The system must support user-generated questions, threaded replies, and engagement features such as voting. It must allow categorization of questions and support filtering by status (active, answered, resolved). The system should provide a scalable backend with a relational database, RESTful API endpoints, and efficient data retrieval. Additionally, the platform should support real-time updates, role-based interactions (students vs instructors), and a responsive frontend interface.

TECH STACK



SYSTEM DESIGN

The system follows a client-server architecture. The frontend is built using React and communicates with a Node.js/Express backend through RESTful APIs. The backend uses Prisma ORM to interact with a PostgreSQL database. Core entities include Users, Courses, Questions, Replies, and Votes. The system is designed to support modular expansion and scalable data handling, with clearly defined routes and structured data relationships.

OBJECT DESIGN

The system is structured around key relational models:

- User: represents students and instructors
- Course: represents academic classes
- Question: stores user-submitted questions with metadata such as category and status
- Reply: stores threaded responses linked to questions
- Vote: tracks user engagement and prioritization

Relationships are enforced using foreign keys, and constraints ensure data integrity (e.g., one vote per user per question).

IMPLEMENTATION

The backend was implemented using Node.js with Express for API routing and Prisma ORM for database management. PostgreSQL was used as the primary database. The system includes fully implemented CRUD operations, validation, error handling, and relational querying. Features such as filtering, sorting by vote count, and status updates were implemented to improve usability. The frontend was developed using React and integrates with the backend through API endpoints.

VERIFICATION



The system was tested using API testing tools such as Thunder Client. Each endpoint was validated using both successful and failure scenarios, including invalid inputs, missing records, and duplicate actions. CRUD operations were verified for all major entities, and relational queries were tested to ensure correct data retrieval. The backend was confirmed to handle edge cases and return appropriate HTTP status codes.

SUMMARY

This project demonstrates the development of a scalable and structured discussion platform for academic environments. By combining a robust backend, relational database design, and interactive features such as voting and filtering, the system improves collaboration and prioritization of student questions. Future work includes real-time updates, authentication, and full frontend integration.

REFERENCES

Node.js Documentation
Express.js Documentation
Prisma ORM Documentation
PostgreSQL Documentation
React Documentation

ACKNOWLEDGEMENT

We'd like to thank Professor Masoud Sadjadi for his assistance and mentorship throughout the development of ClassConnect.